Scalable Strategies for Large-scale AC-SCOPF Problems

Nai-Yuan Chiang[1] Victor M Zavala[1] Andreas Grothey[2]

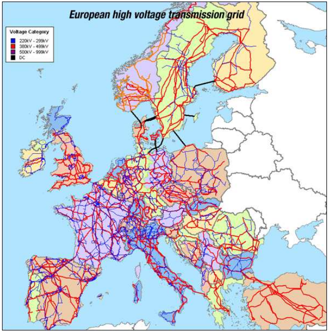
[1]Mathematics and Computer Science, Argonne National Laboratory

[2]School of Mathematics, University of Edinburgh

25 June 2013

Motivation

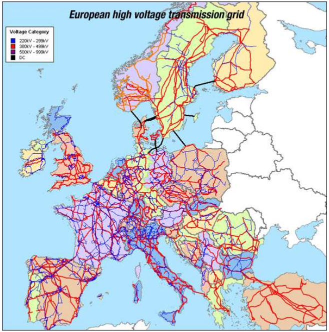
How to exploit structure in power grid problems?



Motivation

How to exploit structure in power grid problems?

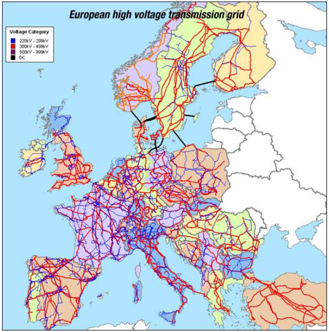
- What tools?



Motivation

How to exploit structure in power grid problems?

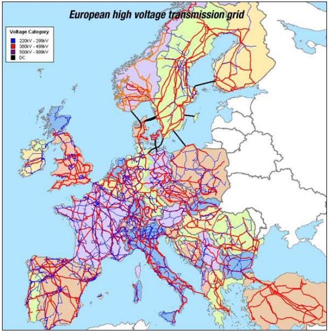
- What tools?
  1) Interior Point Methods
  2) Parallel Linear Algebra
  3) Iterative Solver



Motivation

How to exploit structure in power grid problems?

- What tools?
    1) Interior Point Methods
    2) Parallel Linear Algebra
    3) Iterative Solver
- Applications



Interior Point Methods (IPM)

Nonlinear Program

$$\min \mathbf{f}(x) \quad \text{s.t.} \quad \begin{array}{rcl} \mathbf{c}(x) & = & 0 \\ x & \geq & 0 \end{array} \tag{NLP}$$

KKT Conditions

$$\begin{array}{rcl} \nabla \mathbf{f}(x) - \nabla \mathbf{c}(x)\lambda - s & = & 0 \\ \nabla \mathbf{c}^\top x & = & 0 \\ XSe & = & 0 \\ x, s & \geq & 0 \end{array} \tag{KKT}$$

$X = \mathrm{diag}(x), S = \mathrm{diag}(s)$

Interior Point Methods (IPM)

Barrier Problem

$$\min \mathbf{f}(x) - \mu \sum \ln x_i \quad \text{s.t.} \quad \begin{aligned} \mathbf{c}(x) &= 0 \\ x &\geq 0 \end{aligned} \qquad (\text{NLP}_\mu)$$

KKT Conditions

$$\begin{aligned} \nabla \mathbf{f}(x) - \nabla \mathbf{c}(x)\lambda - s &= 0 \\ \nabla \mathbf{c}^\top x &= 0 \\ XSe &= \mu e \\ x, s &\geq 0 \end{aligned} \qquad (\text{KKT}_\mu)$$

$X = \mathrm{diag}(x), S = \mathrm{diag}(s)$

- Introduce logarithmic barriers for $x \geq 0$
- For $\mu \to 0$ solution of (NLP_μ) converges to solution of (NLP)
- System (KKT_μ) can be solved by Newton's Method

Newton-Step in IPM

Newton-Step: Augmented System(IPM)

$$\Phi = \begin{bmatrix} -H - \Theta & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \xi_c - X^{-1} r_{xs} \\ \xi_b \end{bmatrix}$$

where $\mathcal{A}$ is the constraint Jacobian, and H is the Hessian of the Lagrangian function L.

- NLP needs more work to ensure global convergence.
- IPM with filter technique (IPOPT[1]).

[1] Andreas Wächter and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Math. Program.* 106.1, Ser. A (2006), pp. 25–57. ISSN: 0025-5610.

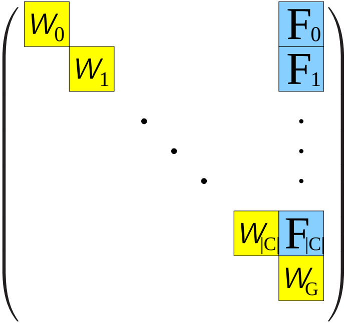
Parallel Linear Algebra for IPM

Newton-Step: Augmented System(IPM)

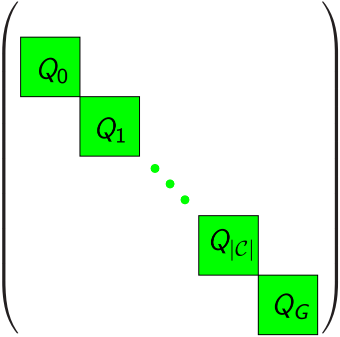
$$\Phi = \begin{bmatrix} -\Theta & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \xi_c - X^{-1} r_{xs} \\ \xi_b \end{bmatrix}$$

where $\Theta = X^{-1}S$, $X = diag(x)$, $S = diag(s)$

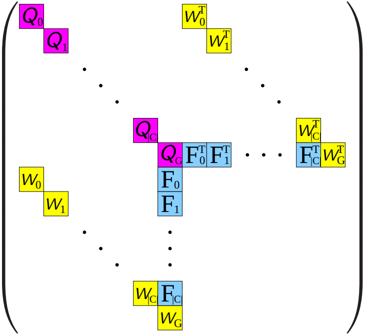
Matrix $\mathcal{A}$



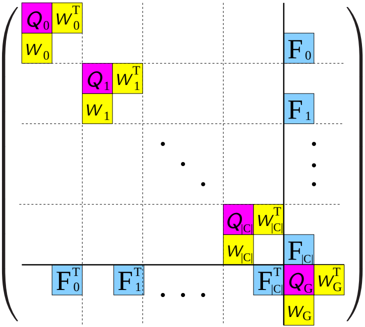
Matrix -Θ



Structures of $\mathcal{A}$, Q and Φ:

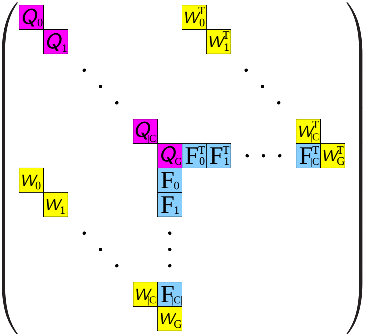


$$\begin{pmatrix} Q & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{pmatrix}$$

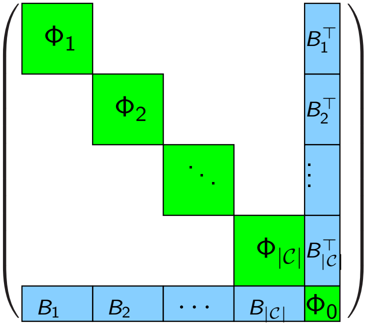


$$P \begin{pmatrix} Q & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{pmatrix} P^{-1}$$

Structures of $\mathcal{A}$, Q and Φ:



$$\begin{pmatrix} Q & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{pmatrix}$$



$$P \begin{pmatrix} Q & \mathcal{A}^\top \\ \mathcal{A} & 0 \end{pmatrix} P^{-1}$$

Bordered block-diagonal structure in Augmented System!

Exploiting Structure in IPM

Block-Factorization of Augmented System Matrix

$$\underbrace{\begin{pmatrix} \Phi_1 & & & B_1^\top \\ & \ddots & & \vdots \\ & & \Phi_n & B_n^\top \\ B_1 & \cdots & B_n & \Phi_0 \end{pmatrix}}_{\Phi} \underbrace{\begin{pmatrix} x_1 \\ \vdots \\ x_n \\ x_0 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \\ \mathbf{b}_0 \end{pmatrix}}_{\mathbf{b}}$$

Solution of Block-system by Schur-complement

The solution to $\Phi x = \mathbf{b}$ is

$$\begin{aligned} x_0 &= C^{-1}\mathbf{b}_0, \quad \mathbf{b}_0 = b_0 - \textstyle\sum_i B_i \Phi_i^{-1} \mathbf{b}_i \\ x_i &= \Phi_i^{-1}(\mathbf{b}_i - B_i^\top x_0), \qquad i = 1, \ldots, n \end{aligned}$$

where C is the *Schur-complement*

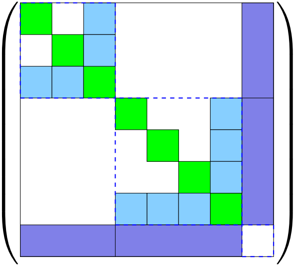
$$C = \Phi_0 - \sum_{i=1}^{n} B_i \Phi_i^{-1} B_i^\top$$

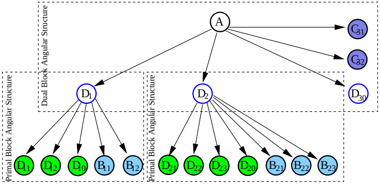
$\Rightarrow$ only need to factor Φ_i, not Φ

Paraller Linear Algebra for the Structured Problem

Parallel IPM Implementation

- Jacek Gondzio and Andreas Grothey: Exploiting structure in parallel implementation of interior point methods for optimization.
- Cosmin G. Petra and Mihai Anitescu: A preconditioning technique for Schur complement systems arising in stochastic optimization.





Paraller Linear Algebra for the Structured Problem

Structure comes from ...

- Robust Stochastic Programming (scenarios)
- Network (partitions)

Paraller Linear Algebra for the Structured Problem

Structure comes from ...

- Robust Stochastic Programming (scenarios)
- Network (partitions)

- Still computationally expensive: memory and communication

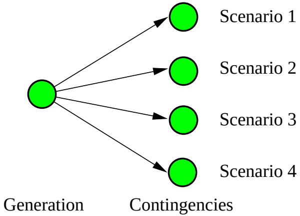
Paraller Linear Algebra for the Structured Problem

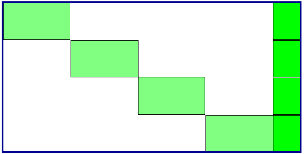
Structure comes from ...

- Robust Stochastic Programming (scenarios)
- Network (partitions)

- Still computationally expensive: memory and communication
- Possible remedies:
  - a) scenario elimination
  - b) iterative method

Scenario Elimination





Scenario Elimination

- Start from a smaller model with one "base" scenario. (e.g the OPF problem)
- Generate a central point for the reduced problem.
- Fix the global variables and find feasible solutions of other scenarios. (Contingency Analysis)
- Add violated scenario dynamically.

Scenario Elimination

Algorithm:

Initialize the active scenario set with the base scenario
In each IPM iter:
Set up the model with all the active scenarios
Solve reduced model to obtain the first stage variables
repeat
  Solve inactive scenarios
  Check for violated contingency scenarios
  Add violated scenarios to the active scenario set
  Re-solve model to obtain new first stage variables
until no more violated contingencies

Scenario Elimination

Algorithm:

Initialize the active scenario set with the base scenario
In each IPM iter:
Set up the model with all the active scenarios
Solve reduced model to obtain the first stage variables
repeat
  Solve inactive scenarios
  Check for violated contingency scenarios
  Add violated scenarios to the active scenario set
  Re-solve model to obtain new first stage variables
until no more violated contingencies

Warmstart

- The above scheme results in a series of models each with an increasing number of binding scenarios.

Applications

Applications

- AC-SCOPF: Scenario Elimination
- DC-SCOPF: Iterative Method + Scenario Elimination(*)
- DC-OPF: Network Partition

AC-SCOPF: Scenario Elimination

Generic AC OPF Model

Optimal Power Flow (OPF)

A minimum cost power generation model.

Parameters

α_l, β_l	conductance and susceptance of line l
β_b	susceptance of power source at bus b
d_b^P, d_b^Q	real and reactive power demand at bus b
f_l^+	flow limit for line l

Variables

v_b	Voltage level at bus b
δ_b	Phase angle at bus b
$p_g,\ q_g$	Real and reactive power output at generator g
$f_{(i,j)}^P, f_{(i,j)}^Q$	Real and reactive power flow on line $l = (i,j)$

Generic AC OPF Model

Constraints

- Kirchhoff's Voltage Law (KVL)

$$\begin{aligned} f^P_{(i,j)} &= \alpha_l v_i^2 - v_i v_j[\alpha_l \cos(\delta_i - \delta_j) + \beta_l \sin(\delta_i - \delta_j)] \\ f^Q_{(i,j)} &= -\beta_l v_i^2 - v_i v_j[\alpha_l \sin(\delta_i - \delta_j) - \beta_l \cos(\delta_i - \delta_j)] \end{aligned}$$

- Kirchhoff's Current Law (KCL)

$$\begin{aligned} \sum_{g|o_g=b} p_g &= \sum_{(b,i)\in L} f^P_{(b,i)} + d^P_b, \quad \forall b \in \mathcal{B} \\ \sum_{g|o_g=b} q_g - \beta_b v_b^2 &= \sum_{(b,i)\in L} f^Q_{(b,i)} + d^Q_b, \quad \forall b \in \mathcal{B} \end{aligned}$$

- Line Flow Limits at both ends of each line

$$\begin{aligned} (f^P_{(i,j)})^2 + (f^Q_{(i,j)})^2 &\le (f_l^+)^2 \\ (f^P_{(j,i)})^2 + (f^Q_{(j,i)})^2 &\le (f_l^+)^2 \end{aligned}$$

- Reference bus

$$\delta_0 = 0$$

$\Rightarrow$ AC OPF is a nonlinear programming problem

Security-Constrained Optimal Power Flow (SCOPF)

(N-1)SCOPF

Network should survive the failure of any one line (possibly after limited corrective actions) without line-overloads.

Setup

- Contingency scenarios $c \in \mathcal{C}$, each has its own power transmission network.
- Real generation p_g and Voltage v_g at the PV bus keep same for all contingencies. (Global Variables)
- Each contingency has its flow, voltage, phase angle and reactive generation: $f_c^{P/Q}, v_c, \delta_c, q_c$. (Local Variables)
- Possible modification of generator output p_c in each contingency scenario.
- Seek a generator setting that does not create line overloads for any contingency

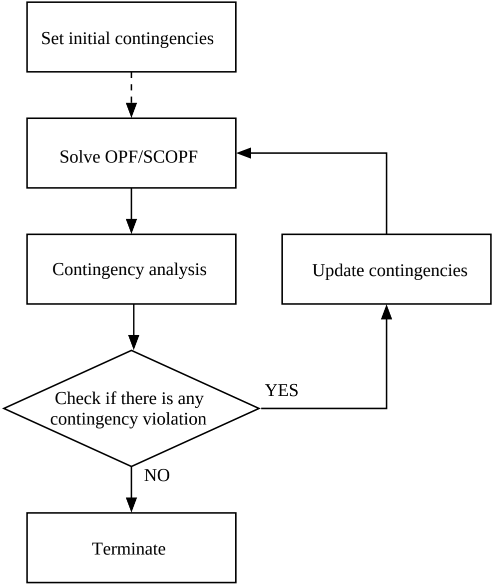
Structure of the Problem

$$\begin{pmatrix} P_0 & F_0 \end{pmatrix} \qquad \begin{pmatrix} P_0 & & & & F_0 \\ & P_1 & & & F_1 \\ & & \ddots & & \vdots \\ & & & P_{|C|} & F_{|C|} \end{pmatrix}$$

OPF SCOPF

- SCOPF (like many other structured problems) consists of a small core that is repeated many times.
- "n-1" requires the inclusion of many contingency scenarios.
- Only a few contingencies are critical for operation of the system (but which ones)?

Flow Chart for Solving SCOPF: State of the Art



Structured IPM with Scenario Elimination

Advantages: (From the Engineering Side)

- Only a few contingencies are critical for operation.
- Start from solving much smaller problem of same structure, as the practical SCOPF solution technique.

Advantages: (From the Mathematical Side)

- Total number of linear algebra in each IPM iteration is proportional to the size of problem.
- IPM is an iterative solution technique. Each IPM step generate a central point for the barrier problem with the given barrier parameter μ.

Apply contingency analysis between IPM steps.
Combine two iterative processes in one.

Numerical Result

		Original			Scenario Elimination		
Prob	No.Sce	time(s)	iters	No.Act	time(s)	iters	No.ActSce
A	1	<0.1	9	0	<0.1	9	1
B	2	<0.1	22	0	<0.1	9	1
6	2	<0.1	13	2	<0.1	13	2
IEEE_24	38	5.7	41	6	3.9	30	6
IEEE_48	78	51.8	71	11	32.4	52	15
IEEE_73	117	204.1	97	16	156.7	92	25
IEEE_96	158	351.5	106	20	252.9	76	27
IEEE_118	178	???	??	42	1225.2	75	46
IEEE_192	318	2393.7	132	26	1586.0	92	40
L26	41	0.4	14	2	0.3	11	2
L200	371	264.3	53	7	56.4	25	7
L300	566	1153.1	88	17	196.3	22	20

Table: Scenario elimination results

- More than 200% computational resources are saved! (Small examples are shown in the end.)

DC-SCOPF: Iterative Method + Scenario Elimination(*)

Structure of DC OPF problem

Given

- bus/generator incidence matrix $J \in \boldsymbol{R}^{|\mathcal{B}| \times |\mathcal{G}|}$
- node/arc incidence matrix $A \in \boldsymbol{R}^{|\mathcal{B}| \times |\mathcal{L}|}$
- $R = \mathrm{diag}(-v^2/r_1, \ldots, -v^2/r_{|\mathcal{L}|}), D = \sum_b d$

DC-OPF problem can be written as

DC-OPF

$$
\begin{array}{llllll}
\min & c^\top p_g & & & & \\
\text{s.t.} & Rf & +A^\top \delta & & = 0 \\
 & Af & & -Jp_g & = -d \\
 & & & e^T p_g & = D
\end{array}
$$

$\Rightarrow$ DC OPF is a linear/quadratic programming problem

Security-Constrained Optimal Power Flow (SCOPF)

DC-OPF

$$
\begin{array}{lllll}
\min & c^\top p_g & & & \\
\text{s.t.} & Rf & +A^\top \delta & & = 0 \\
& Af & & & = Jp_g - d \\
& & & e^T p_g & = D
\end{array}
$$

DC-SCOPF

$$
\begin{array}{llllll}
\min & c^\top p_g & & & & \\
\text{s.t.} & Rf_c & +A_c^\top \delta_c & & = 0, & \forall c \in \mathcal{C} \\
& A_c f_c & & & = Jp_g - d, & \forall c \in \mathcal{C} \\
& & & e^\top p_g & = D &
\end{array}
$$

Iterative Method: GMRES

Bottleneck in this process

- Assembling Schur-complement $C = \Phi_0 - \sum_{i=1}^{n} B_i \Phi_i^{-1} B_i^\top$ is very expensive!
- Get the solution to $Cx_0 = \mathbf{b}_0$ without having C explicitly!

Iterative Method: GMRES

Bottleneck in this process

- Assembling Schur-complement $C = \Phi_0 - \sum_{i=1}^{n} B_i \Phi_i^{-1} B_i^\top$ is very expensive!
- Get the solution to $Cx_0 = \mathbf{b}_0$ without having C explicitly!

$\Rightarrow$ Solve $Cx_0 = \mathbf{b}_0$ by iterative method

- Use (preconditioned) iterative method (e.g. GMRES)
- with $M = \Phi_0 + nB_0\Phi_0^{-1}B_0^\top$ as preconditioner for SCOPF (Qiu, Flueck '05)

$\Rightarrow$ Evaluating residuals $r = \mathbf{b}_0 - Cx_0$ is via:

$$Cx_0 = \Phi_0 x_0 + \sum_{i=1}^{n} B_i \Phi_i^{-1} B_i^\top x_0.$$

Scenarios Elimination for Preconditioner

"Active contingencies"

- Some scenarios may have very large entries in its contribution of Schur complement
  (small slack variable)
- This slack variable is small through the whole IPM process.

These scenarios are critical!

How to choose a preconditioner

- "Aggressive" method:
  reset the preconditioner in each IPM iteration.
- "Cumulative" method:
  keep the scenario in the preconditioner till the end of IPM process.

Summary of the Test Problems

buses	contingencies	variables	constraints	nonzeros(%)
3	2	17	14	14.7059
26	40	2,630	2,626	0.11481
56	79	10,648	10,642	0.02741
100	180	50,344	50,320	0.00654
200	370	210,779	210,730	0.00158
300	565	488,534	488,460	0.00069
400	760	881,339	881,240	0.00038
500	955	1,389,194	1,389,070	0.00024

Table: Summary of test problems

Numerical Results

		Cumulative			Aggressive		
Bus	NoSce	Time(s)	Iters	FinSce	Time(s)	Iters	MaxSce
3	3	<0.1	8	2	<0.1	8	1
26	41	0.27	13	2	0.27	13	1
56	80	1.26	15	6	1.25	15	4
100	181	9.09	20	7	9.08	20	6
200	371	50.63	28	9	50.16	28	7
300	566	205.79	39	20	234.70	43	19
400	761	523.65	55	20	529.48	56	16
500	956	823.95	46	25	823.91	47	21

Table: GMRES with different methods to build preconditioners

The number of dominant scenarios is less than 5% of the number of total scenarios!

Numerical Results

	Direct Method		Cumulative		Aggressive	
buses	time(s)	memory	time(s)	memory	time(s)	memory
3	$<$0.1	5.2MB	$<$0.01	5.2MB	$<$0.01	5.2MB
26	0.17	7.5MB	0.27	7.4MB	0.27	7.4MB
56	0.77	14.1MB	1.26	13.5MB	1.25	13.5MB
100	6.16	53.1MB	9.09	43.6MB	9.08	43.6MB
200	45.23	244MB	50.63	163MB	50.16	163MB
300	177.39	667MB	205.79	387MB	234.70	387MB
400	655.50	1380MB	523.65	715MB	529.48	716MB
500	1195.77	2467MB	823.95	1163MB	823.91	1164MB

Table: Comparisons among three methods

For large problems: Faster & With less memory usage!

DC-OPF: Network Partition

Another Scalable Strategy for Parallelism

Idea: Decompose the model by the power system behavior

- Graph partitioning technique.
- Decompose the large network into several "equal-sized" pieces.
- Minimize the number of edge cuts between separated components.

- Advantages: Solve the model for each piece of cake in parallel!
- Difficulties: Unusual as generic stochastic programming: Partitioning may introduce high degree of coupling vars and constants.

DC-OPF formulation

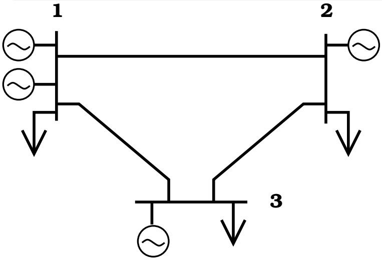
DC-OPF formulation (Default)

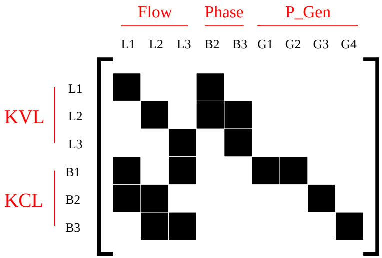
- Kirchhoff's Voltage Law

$$f_l^P = -\frac{v^2}{r_l} \sum_{b \in \mathcal{B}} a_{bl} \delta_b, \quad \forall l \in \mathcal{L}$$

- Kirchhoff's Current Law

$$\sum_{g | o_g = b} p_g = \sum_{(b,i) \in \mathcal{L}} f_{(b,i)}^P + d_b^P, \quad \forall b \in \mathcal{B}$$





The structure of the matrix components in IPM

- Define set $\mathcal{K}$: set of partitions.
- Define set $\mathcal{L}_{cut}$: set of line cuts, $\mathcal{L}_{cut} \subseteq \mathcal{L}$.
- Define $\mathcal{L}_k$: transmission lines in partition k.
- Define $\mathcal{B}_k$: buses in partition k.

DC-OPF formulation with network partition

- KVL for each partition

$$f_l^P = -\frac{v^2}{r_l} \sum_{b \in \mathcal{B}} a_{bl} \delta_b, \quad \forall l \in \mathcal{L}_k, \quad \forall k \in \mathcal{K}$$

- KCL for each partition

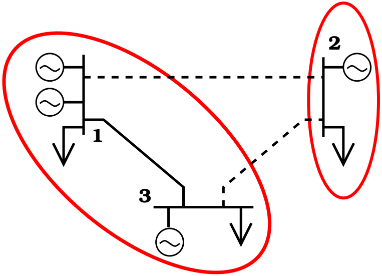
$$\sum_{g | o_g = b} p_g = \sum_{(b,i) \in \mathcal{L}} f_{(b,i)}^P + d_b^P, \quad \forall b \in \mathcal{B}_k, \quad \forall k \in \mathcal{K}$$

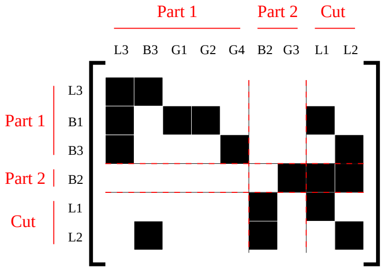
- KVL for the cuts

$$f_l^P = -\frac{v^2}{r_l} \sum_{b \in \mathcal{B}} a_{bl} \delta_b, \quad \forall l \in \mathcal{L}_{cut}$$

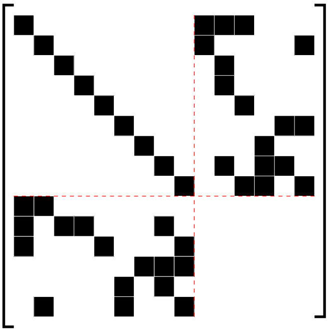
The structure of the matrix components in IPM

- Each partition corresponds to a diagonal block in the constraint Jacobian.
- Variables and constraints corresponding to the cuts are moved to the borders.

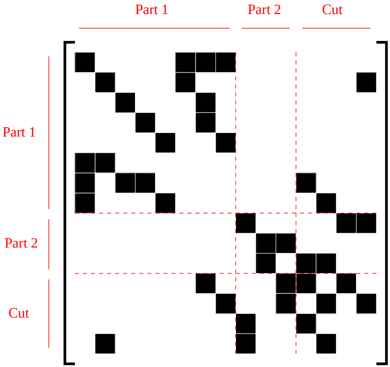




Structures of the Augmented System:

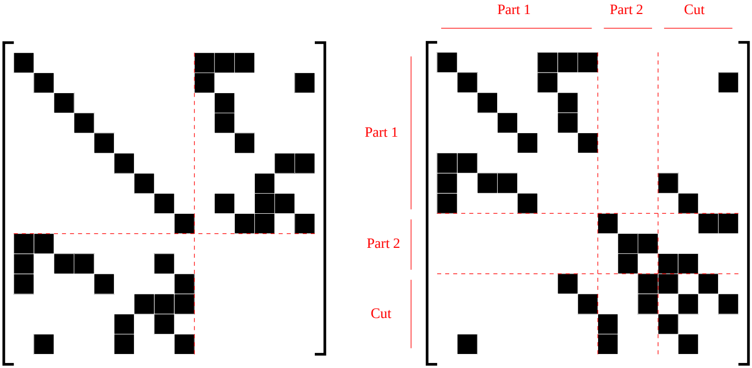


Augmented System



Reordered Augmented System

Structures of the Augmented System:



Augmented System   Reordered Augmented System

- The size of Schur complement is 2 times #.cuts!

The Illinois system

How does the network partition look like for the real system?

The Illinois system

How does the network partition look like for the real system?

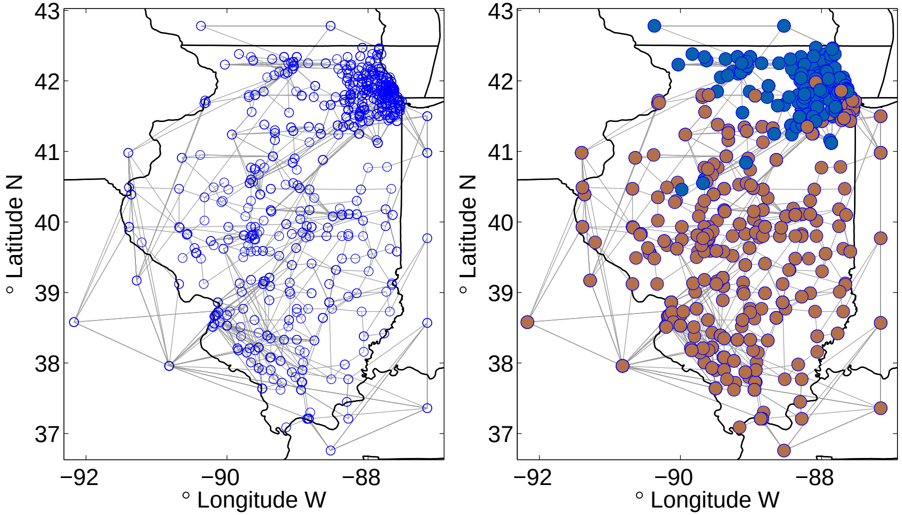
- Illinois system: 1908 buses and 2522 lines

The Illinois system

How does the network partition look like for the real system?

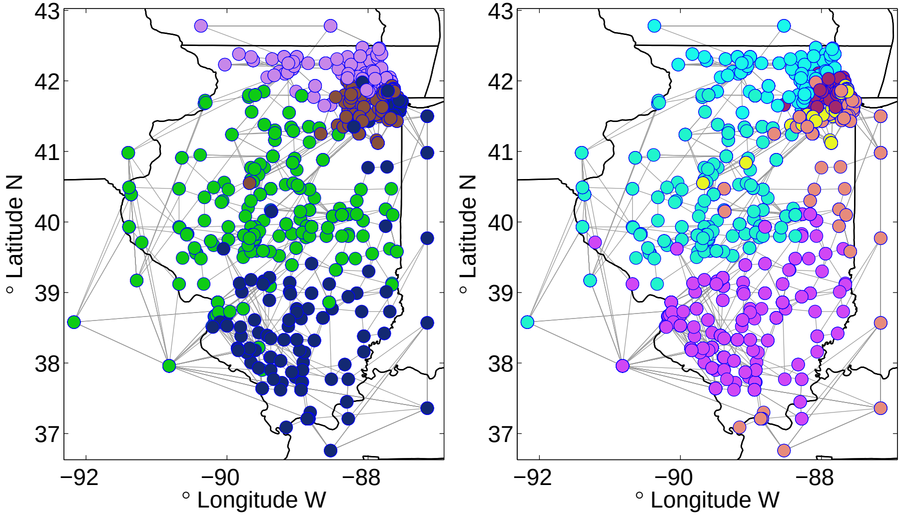
- Illinois system: 1908 buses and 2522 lines
- Is network partition obvious?
- How many coupling variables and constraints will be introduced?
- How would this affect the computational scalability?
- What number of partitions is sensible to apply?

The Illinois system



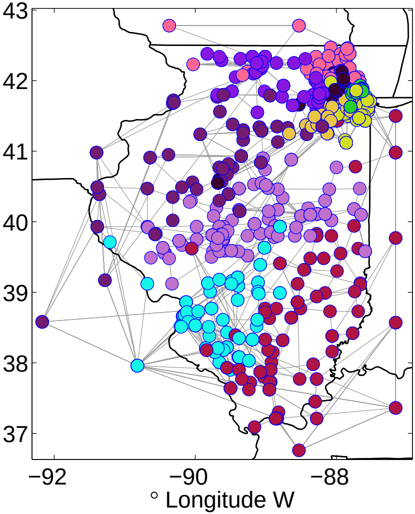
Illinois system and the system with 2 partitions.

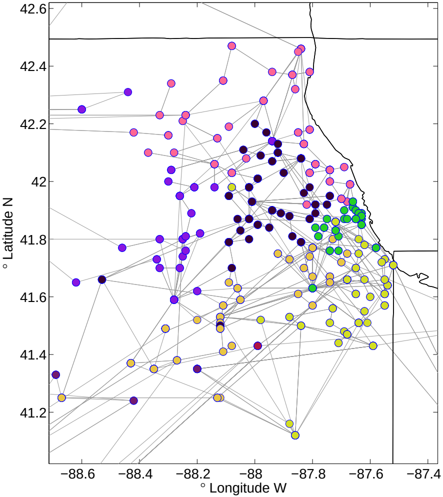
The Illinois system



Illinois system with 4 and 6 partitions.

The Illinois system



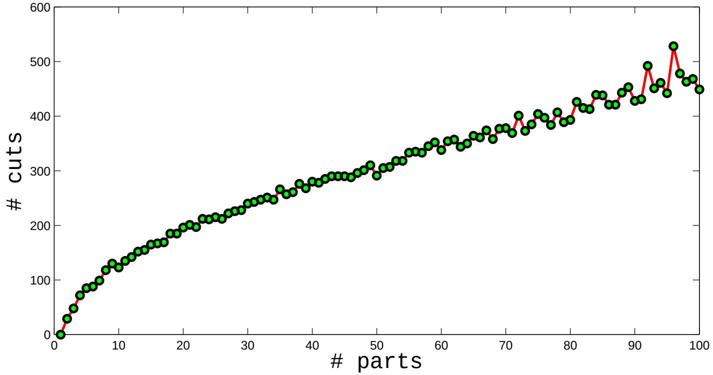


Illinois system with 10 partitions.

The Illinois system

Coupling constraints = # of the edge-cuts

- Determine the size of the Schur complement.
- Communication between processes! Parallel efficiency!



Numerical results from prototype

- 4690 variables and 4430 constraints $\rightarrow$ one time slot

Illinois system: DC-OPF

- Network partition:
  less than 0.1s for network partitions (by Metis), regardless of the number of partitions (from 1 to 100).
  Each part only contains 20 buses (with 100 partitions)!
- Solution time:
  4 partitions with four processes (72 cuts):
  a) faster than solving the problem in serial.
  100 partitions with four processes (449 cuts):
  b) slower than solving the problem in serial. (Only 191 buses in each part, but the size of Schur complement is large $\rightarrow$ more expensive to solve this problem.)

Scenarios can also be included in the model $\rightarrow$ nested structure.

Conclusions

Problems is complicated

- Illinois system with 24 hours slots and Wind:
  10 mins in serial (CPLEX) for the relaxation of the Unit Commitment.i

Conclusions

Problems is complicated

- Illinois system with 24 hours slots and Wind:
  10 mins in serial (CPLEX) for the relaxation of the Unit Commitment.i

We expect:

- 24 time steps UC for the Illinois system: $\approx$ 2.5 mins in parallel
- Partitioning with 10 parts is appliable: 100 cuts per time slot; $100*24 = 2400$ Coupling variables for the full problem; speed up the solution time by a factor of 10!

Conclusions

Problems is complicated

- Illinois system with 24 hours slots and Wind:
  10 mins in serial (CPLEX) for the relaxation of the Unit Commitment.i

We expect:

- 24 time steps UC for the Illinois system: $\approx$ 2.5 mins in parallel
- Partitioning with 10 parts is appliable: 100 cuts per time slot; 100*24 $=$ 2400 Coupling variables for the full problem; speed up the solution time by a factor of 10!

Future Work: merge all the tools

- Complete the NLP tool to solve the AC stochastic problem (Time! UC/ED! Security!)
- Apply the scenario elimination technique and iterative methods.

Conclusions

- Thank you for your attention!

Two Small Examples:

Objective

$$\min \quad (x-1)^2$$

Constraints A

$$\begin{aligned} \text{s.t.} \quad x^2 + y &= 100 \\ 0 \le x, y &\le 100 \end{aligned}$$

$\Rightarrow$ 9 Iter,

Constraints B

$$\begin{aligned} \text{s.t.} \quad x^2 + y &= 100 \\ x^2 + z &= 100 \\ 0 \le x, y, z &\le 100 \end{aligned}$$

$\Rightarrow$ 22 Iter

Two Small Examples:

Objective

$$\min \quad (x-1)^2$$

Constraints A

$$\begin{aligned} \text{s.t.} \quad x^2 + y &= 100 \\ 0 \le x, y &\le 100 \end{aligned}$$

$\Rightarrow$ 9 Iter,

Constraints B

$$\begin{aligned} \text{s.t.} \quad x^2 + y &= 100 \\ x^2 + z &= 100 \\ 0 \le x, y, z &\le 100 \end{aligned}$$

$\Rightarrow$ 22 Iter $\Rightarrow$ 9 Iter!

- Only pay attention to the useful scenarios
- Smaller binding problem = Less numerical difficulties